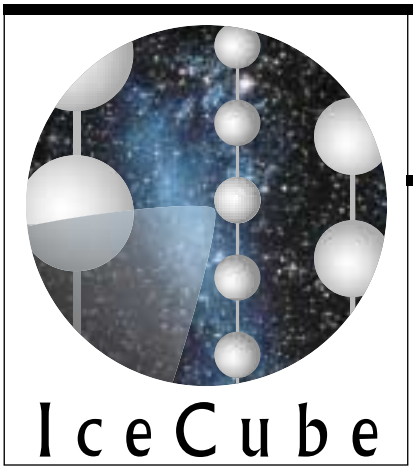



# IceCube Software

# *The Software Design Description of DAQ*

**Chuck McParland**

Last Modified: October 24, 2002 3:30 pm

**Version:**0.0

# 1.0 Introduction

## 1.1 Purpose

The purpose of this document is to provide an overall description of the IceCube data acquisition system. This document should serve as a general introduction to the major data acquisition system components, their behavioral relationship to each other, and their relationship to other IceCube software subsystems. Detailed descriptions of these system components will be found in subsequent documents.

## 1.2 Scope

As noted above, this document will focus on the major software components of the IceCube data acquisition system. In addition to interacting with each other, these components also interact with design entities outside the scope of this document. These entities (e.g. "logging services") will be described in other documents.

## 1.3 References

IceCube Preliminary Design Document

Software Requirements Specification for IceCube System Software

DAQ System Requirements Document

# 2.0 Background

## 2.1 Need for Robustness

By its very nature, the IceCube experiment requires a data acquisition system that is capable of dealing with a large number of independent sensors operating autonomously in a remove environment. The experiment will deploy 4860 optical modules organized into 81 strings over a period of 5 to 6 years at a location justifiably described as "at the end of the earth." Once operating, it will produce approximately 12 Gbytes of data daily and close to 4 Tbytes of data per year. Data will be transferred daily via satellite connection to the northern hemisphere where archiving and final analysis will occur. For 9 months out of the year, during the winter season, the system will have only minimal local support and many operations will be remotely initiated and monitored via satellite link. Furthermore, the entire activity is expected to continue for at least 10 years after deployment. If ever there were a "poster project" for robust, reliable system design, IceCube would be an excellent candidate.

## 2.2 Performance

Data rates flowing from individual DOMs within the detector array are relatively modest. However, when aggregated into an array of roughly 5000 contributing data sources, they create significant network demands for both in terms of raw data transport bandwidth and data messaging traffic. The system design addresses these requirements through two mechanisms. First, the system is organized as a hierarchy consisting of

groups of DOMs that share a common deployment infrastructure-the deployment string. Thus, the data acquisition system sees the detector as a collection of logical strings each of which contains a set of individual DOMs. This hierarchical organization is mirrored in the physical arrangement of both hardware and software components. As a result, the primary data path taken by individual DOM "hits" as they are combined into IceCube events is well understood and the system design can be optimized to accommodate it.

However, in addition to this primary data path, the data acquisition system must also gather hit information from all DOMs into a single process in order to make event trigger decisions based on detector-wide response. This necessitates message traffic between software components that does not follow the same hierarchical data paths used by the bulk data flow. This traffic pattern is accommodated by the use of efficient, switched data path networking fabrics that allow direct connection of all DOM strings to both the event builder and global trigger software components. These commercial network switches allow us to effectively distribute data acquisition system software components across multiple CPUs in a processor farm.

## 2.3 Incremental Deployment Schedule

As noted earlier, the full detector array will take a number of years to fully deploy. Furthermore, actual deployment activities will only take place during the 3+ months of the austral summer season. As a result, large, but incomplete, portions of the detector will be available for use several years prior to the completion of the detector. IceCube detector operations expect continuous data acquisition from all detector elements from the time of their deployment. Therefore, the data acquisition system design must, from the very beginning, accommodate periodic deployment of new detector elements and operation of an incomplete detector. It must also be able to integrate new detector elements with a minimum amount of downtime. The ability to scale system software, computing hardware and network infrastructure as new detector elements are added is a clear requirement for both the software design and the hardware installation on which it runs.

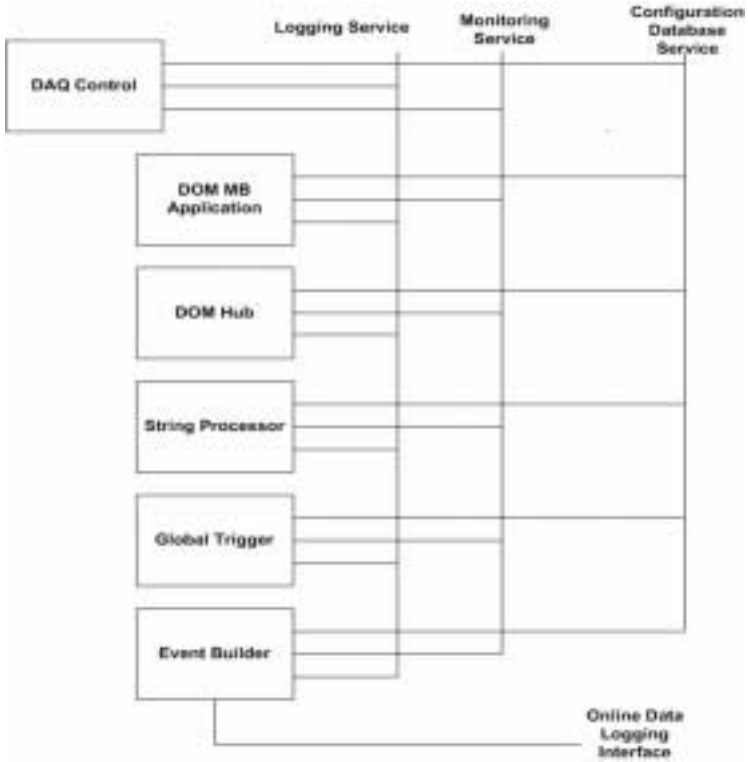
## 2.4 Use of generic hardware and software platforms

These system characteristics have lead us to a data acquisition system design that relies heavily on generic computing platforms and commercial networking infrastructure. While both the DOM and its specialized surface communications interface are clearly unique to the IceCube experiment, the design of the remaining software system elements depends only on the availability of generic computing platforms and networking capabilities. During implementation, software components can be aggregated onto single processors or distributed across multiple CPUs depending on performance requirements. Although we anticipate using identical CPUs as processor farm building blocks, the design allows some functions to be assigned to high performance processors as needed without major changes to the underlying system architecture or implementation. While some efficiencies of size or power might be achieved by denser system packaging (e.g. Compact PCI) or faster interprocessor message passing performance (e.g. VME backplane or Myranet), implementations using these solutions would tightly couple the hardware and software and bind it to a particular hardware platform. Therefore, in the following design description, there will be little discussion of specific processor or networking platforms.

## 3.0 Decomposition View

The IceCube DAQ is divided into the following design entities (see figure 1):

**FIGURE 1.**




### 3.1 DOM MB Application

**ID:** DOM MB Application

**Type:** Subsystem

**Purpose:** The DOM MB application is the DOM-resident software that configures and controls all data taking behavior internal to each DOM. It is responsible for acquiring data produced by digitizing PMT output waveforms and communicating them, on demand, to the DOM Hub subsystem.

**Function:** DOM MB Application provides:

- Control of all DOM-resident, software controllable hardware functions.
- Transfer, storage and execution control of all software programs to be executed on the DOM MB.
- Transfer, storage and loading of all programable firmware used within the DOM MB programable logic.

- Control of DOM-resident waveform digitization, compression and data buffering functions.
- Control communications and transport of acquired data to surface subsystem (i.e. DOM Hub).

**Subordinates:** none

### 3.2 DAQ Control

**ID:** DAQ Control

**Type:** Subsystem

**Purpose:** DAQ Control provides a single control view of all DAQ components. As such, it provides a single API through which higher software and operator levels can determine the overall state of the DAQ system and command that system to move to one of a small set of known operational states.

**Function:** DAQ Control provides

- Single access point of control and monitoring of overall DAQ state and operation.
- Control and sequencing of all individual DAQ components to achieve overall operational state.
- Periodic monitoring of individual DAQ components to assure their proper state.
- Automated DAQ-wide response to detected error conditions.

**Subordinates:**

- DOM MB Application
- DOM Hub
- String Processor
- Global Trigger
- Event Builder
- Configuration Database
- Logging Service
- Online Data Logging Interface

### 3.3 DOM Hub

**ID:** DOM Hub

**Type:** Subsystem

**Purpose:** The DOM Hub is the sole electrical attachment point for all deployed IceCube DOMs. As such it must provide power to all DOMs and control and mediate all communications between DAQ subsystems and individual DOMs. Since DOM timebase calibrations, which are performed on a continual periodic basis, must be carefully co-ordinated with all surface to DOM communications, the DOM Hub is also responsible for managing and performing these calibration activities. In addition, the

DOM Hub is responsible for continually querying all attached and operational DOMs for any data contained in their data buffers.

**Function:** The DOM Hub provides

- Communications nexus for all DOMs attached to a given hub.
- Management of power and operational state for all attached DOMs. This includes control of boot sequences to achieve execution of test programs as well as the DOM MB application.
- Control nexus for all requests to change either DOM operational state or configuration (i.e. "slow control").
- Continuous data collection from all attached and operational DOMs.
- Periodic time calibration of DOM MB time base with respect to internal DOM Hub time base and UTC time.
- Data forwarding service that forwards data collected from each operational DOM to the appropriate String Processor.
- Forwarding, to the appropriate String Processor, of periodic time base calibrations performed on each DOM.
- Forwarding, to the appropriate String Processor, of periodic DOM configuration descriptions obtained for each operational DOM.
- Collect subsystem specific performance monitoring information and periodically present such information to the logging service.

**Subordinates:** DOM MB Application

### 3.4 String Processor

**ID:** String Processor

**Type:** Subsystem

**Purpose:** The string processor is the first subsystem within the data flow from DOM to storage that has immediate access to data from all operational DOMs on a particular detector string for a known time interval. By use of periodic time calibration data collected by the DOM Hub, it is responsible for normalization of individual time tags presented by each DOM into a UTC time representation. Once normalized to UTC, the string processor is responsible for locating significant clusters of PMT signals in both the time and string location dimension and presenting a synopsis of those hits to the global trigger subsystem. When asynchronously directed by the global trigger to forward DOM data for a given time interval, it gathers data for that interval from all contribution DOMs and forwards it to the event builder for subsequent processing and recording.

**Function:** String Processor functions

- Data buffering of all DOM data for an entire string for extended periods-up to 30sec.
- Application of individual time calibrations to each operational DOM data stream to create correct UTC time stamp for each data element.
- Periodic extraction of time-ordered, string-local PMT hits into a per string trigger synopsis record.

- Forwarding of each trigger synopsis record produced to the global trigger subsystem.
- When requested by the global trigger subsystem, forward all string-local data within a specified UTC time interval to the event builder subsystem.
- Collect subsystem specific performance monitoring information and periodically present such information to the logging service.

**Subordinates:** none

### 3.5 Global Trigger

**ID:** Global Trigger

**Type:** Subsystem

**Purpose:** The global trigger is primarily responsible for locating detector-wide hit patterns that represent possibly interesting events suitable for further filtering. It takes as its raw material, the trigger synopsis data for a particular time interval provided by all string processors active within the current configuration. These trigger synopses are examined with regard to several event criteria. When any of these event criteria are met, a suitable UTC time interval is calculated to cover all data involved in selected event(s). This interval is sent to all active string processors and to the event builder. Subsequent event processing is performed by those subsystems.

**Function:** Global Trigger provides

- Collect trigger synopsis records from all active string processors.
- Once all trigger synopsis contributions for a given time interval have been collected, search for detector-wide events that meet one or more set of criteria.
- Once an event has been found, communicate TUTC time interval of interest to all active string processors.
- Notify event builder of pending event to be received, built and recorded.
- Collect subsystem specific performance monitoring information and periodically present such information to the logging service.

**Subordinates:** none

### 3.6 Event Builder

**ID:** Event Builder

**Type:** Subsystem

**Purpose:** .The event builder is dedicated to collecting contributions for a given event from all active string processors, create a single, properly formatted data structure to represent that data and pass it to the on-line data logging interface. The event builder performs these functions with no regard to content and acts solely at the direction of the global trigger.

**Function:** Event Builder provides

- Receive requests to build events from the global trigger.
- Receive and synchronize event contributions from all active string processors.
- Collate and reformat individual string processor contributions into a single event data structure.
- Pass built events to the online data logging interface.
- Collect subsystem specific performance monitoring information and periodically present such information to the logging service.

**Subordinates:** none

## 4.0 Dependency View

### 4.1 DOM MB Application

**ID:** DOM MB Application

**Type:** Subsystem

**Dependencies:**

- none

**Resources:**

- DOM MB and associated hardware (PMT, HV supply, etc.)

### 4.2 DAQ Control

**ID:** DAQ Control

**Type:** Subsystem

**Dependencies:**

- Configuration Database
- Logging Service
- Monitoring Service

**Resources:**

- Designated processor within DAQ processor farm
- DAQ processor farm network infrastructure

### 4.3 DOM Hub

**ID:** DOM Hub

**Type:** Subsystem

**Dependencies:**

- DOM MB Application
- Configuration Database
- Logging Service
- Monitoring Service

**Resources:**

- Attached DOMs
- IceCube designed DOM Communications Interface (DCI) cards.
- DAQ Control
- Designated processor within DAQ processor farm
- DAQ processor farm network infrastructure

### 4.4 String Processor

**ID:** String Processor

**Type:** Subsystem

**Dependencies:**

- DOM Hub
- DAQ Control
- Configuration Database
- Logging Service
- Monitoring Service

**Resources:**

- Designated processor within DAQ processor farm
- DAQ processor farm network infrastructure

### 4.5 Global Trigger

**ID:** Global Trigger

**Type:** Software Component

**Type:** Subsystem

**Dependencies:**

- String Processor
- Event Builder
- DAQ Control
- Configuration Database
- Logging Service

- Monitoring Service

**Resources:**

- Designated processor within DAQ processor farm
- DAQ processor farm network infrastructure

### 4.6 Event Builder

**ID:** Event Builder

**Type:** Subsystem

**Dependencies:**

- String Processor
- Global Trigger
- DAQ Control
- Online Data Logging Interface
- Configuration Database
- Logging Service
- Monitoring Service

**Resources:**

- Designated processor within DAQ processor farm
- DAQ processor farm network infrastructure

## 5.0 Interface View

Detailed interface views will appear in subsequent documents that focus on individual software components (e.g. DOM Hub).

## 6.0 Details View

To be completed